Soft Computing
https://doi.org/10.1007/s00500-020-05300-0

FOCUS

Characterization and computation of ancestors in reaction systems

Roberto Barbuti[1] · **Anna Bernasconi**[1] · **Roberta Gori**[1] · **Paolo Milazzo**[1]

© The Author(s) 2020

Abstract
In reaction systems, preimages and nth ancestors are sets of reactants leading to the production of a target set of products in either 1 or n steps, respectively. Many computational problems on preimages and ancestors, such as finding all minimum-cardinality nth ancestors, computing their size or counting them, are intractable. In this paper, we characterize all nth ancestors using a Boolean formula that can be computed in polynomial time. Once simplified, this formula can be exploited to easily solve all preimage and ancestor problems. This allows us to directly relate the difficulty of ancestor problems to the cost of the simplification so that new insights into computational complexity investigations can be achieved. In particular, we focus on two problems: (i) deciding whether a preimage/nth ancestor exists and (ii) finding a preimage/nth ancestor of minimal size. Our approach is constructive, it aims at finding classes of reactions systems for which the ancestor problems can be solved in polynomial time, in exact or approximate way.

Keywords Reaction systems · Ancestor computation · Computational complexity · Causality relations

1 Introduction

Inspired by natural phenomena, many new computational formalisms have been introduced to model different aspects of biology. Basic chemical reactions inspired the reaction systems, a *qualitative* modeling formalism introduced by Brijder et al. (2011), Ehrenfeucht and Rozenberg (2007). It is based on two opposite mechanisms, namely *facilitation* and *inhibition*. Facilitation means that a reaction can occur only if all its reactants are present, while inhibition means that the reaction cannot occur if any of its inhibitors is present. A rewrite rule of a reaction system (called *reaction*) is hence a triple (R, I, P), where R, I and P are sets of objects representing *reactants*, *inhibitors* and *products*, respectively, of the modeled chemical reaction. A *reaction system* is represented by a set of reactions having such a form, together with a (finite) support set S containing all of the objects that can appear in a reaction. The state of a reaction system consists of a finite set of objects describing the biological entities that are present in the real system being modeled. The presence of an object in the state expresses the fact that the corresponding biological entity, in the real system being modeled, is present in a number of copies as high as needed. This is the *threshold supply* assumption and characterizes reaction systems.

A reaction system evolves by means of the application of its reactions. The threshold supply assumption ensures that different reactions never compete for their reactants, and hence, all the applicable reactions in a step are always applied. The application of a set of reactions results in the introduction of all of their products in the next state of the system.

The main advantages of investigating reaction systems are that they have a clean computational model allowing for precise formal analysis, and they can be considered as reference for other computing system (e.g., Barbuti et al. 2018b; Formenti et al. 2014).

Computational complexity of some problems related to the dynamics of reaction systems has been extensively studied (e.g., in Dennunzio et al. 2015a, b; Formenti et al. 2014; Salomaa 2013a, b). In Dennunzio et al. (2015a, b), the authors introduced the concept of preimage and nth ancestor. Roughly speaking, an nth ancestor is a set of objects that lead to the production of a target set of objects after n evolution steps, while a preimage is a 1st ancestor. The authors studied the complexity of several problems related to nth ancestors by defining reductions between well-known hard problems and the corresponding nth ancestor problem. They proved

Communicated by Miguel A. Vega-Rodriguez.

✉ Roberta Gori
gori@di.unipi.it

[1] Dipartimento di Informatica, Università di Pisa, Pisa , Italy

Published online: 07 September 2020

that finding a minimal size preimage or ancestor, computing their size, or counting them are all intractable problems.

In this paper, we propose a *constructive method* to reason on preimages and nth ancestors. Indeed, we define a formula able to characterize all nth ancestors of a given set of objects. Such a formula is obtained by revising the idea of formula-based predictor introduced in Barbuti et al. (2016a, b, 2017, 2018c, d) for reaction systems that interact with an environment. A formula-based predictor is a logic formula that exactly characterizes all the states leading to a given product in a given number of steps. It enables the study of all causal dependencies of one object from the other objects provided by the environment and therefore enhances previous works on causality in reaction systems (Brijder et al. 2010), system biology (Gori and Levi 2010; Bodei et al. 2013, 2015) and natural computing (Busi 2007).

Following the same approach, here we define an nth ancestor formula that fully characterizes all nth ancestors. Moreover, we define an operator able to compute the formula in polynomial time. The relevance of this complete characterization of all nth ancestors is twofold. Ancestors in a closed reaction system express the causality relations between the objects in the initial state and the products that we can obtain after a given number of steps. Assume you are interested in the production of a certain set of objects, ancestors model all the different initial states that can lead to the production of the set of products of interest. Minimal ancestors model minimal states and our complete characterization is able to detect which are the objects that are really necessary for the production of the objects of interest. Moreover, the Boolean formula can be used to give effective algorithms for solving problems as deciding the existence of nth ancestors or the computation of a minimal size preimage or ancestor. This allows us to study the complexity of the preimage and nth ancestor problems in a constructive way by precisely identifying the step in the solving procedure that is the cause for the exponential cost. In particular, in our context, the computationally expensive step consists in the conversion of the formula to disjunctive normal form (DNF) in order to easily solve ancestor problems. An idea to avoid this step is to devise cases in which the conversion into DNF is not necessary.

To this aim, our investigation proceeds in two different directions. First, we identify classes of reaction systems for which our nth ancestor formula is already in DNF; this is obtained by imposing synctactic restrictions on the form of reactions of the reaction system. Then, we define approximations of the nth ancestor formula that can be computed in polynomial time and that are still useful to reason on nth ancestor problems. This approach is validated by several biological examples of gene regulatory networks.

The present paper enhances and extends (Barbuti et al. 2018a). More precisely, the original contributions of the paper are:

1. a new investigation leading to the formal definition of sets of approximations of the original ancestor formula;
2. the definition of effective operators able to compute such approximations in polynomial time;
3. the introduction of biological examples that show the application of the ancestors and validate our approach based on the computation of approximations.

The paper is organized as follows. Section 2 introduces (closed) reaction systems, preimages and nth ancestors. Section 3 presents some preliminary notions. The definition of nth ancestor formulas is given in Sect. 4 together with an effective operator to compute them. In Sect. 5, we bound the complexity of the nth ancestor formula. Section 6 lists some conditions under which the existence and minimal size of ancestors can be computed in polynomial time. In Sect. 7, we introduce two kinds of approximations of nth ancestor formulas that can be computed in polynomial time. Section 8 reports some conclusions.

2 Closed reaction systems

In this section, we recall the basic definition of reaction system (Brijder et al. 2011; Ehrenfeucht and Rozenberg 2007). Let S be a finite set of symbols, called objects. A *reaction* is a triple (R, I, P) with $R, I, P \subseteq S$, composed of *reactants* R, *inhibitors* I and *products* P. Reactants and inhibitors $R \cup I$ of a reaction are collectively called *resources* of such a reaction, and we assume them to be disjoint ($R \cap I = \emptyset$), otherwise the reaction would never be applicable. The set of all possible reactions over a set S is denoted by $\mathrm{rac}(S)$. Finally, a *reaction system* is a pair $\mathcal{A} = (S, A)$, where S is a finite support set and $A \subseteq \mathrm{rac}(S)$ is a set of reactions.

The state of a reaction system is described by a subset T of the set of objects S. Let $a = (R_a, I_a, P_a)$ be a reaction and T a subset of S. The result $\mathrm{res}_a(T)$ of the application of a to T is either P_a, if T separates R_a from I_a (i.e., $R_a \subseteq T$ and $I_a \cap T = \emptyset$), or the empty set $\emptyset$ otherwise. The application of multiple reactions at the same time occurs without any competition for the used reactants (*threshold supply assumption*). Therefore, each reaction which is not inhibited can be applied, and the result of the application of multiple reactions is cumulative. Formally, given a reaction system $\mathcal{A} = (S, A)$, the result of the application of $\mathcal{A}$ to a set $T \subseteq S$ is defined as $\mathrm{res}_{\mathcal{A}}(T) = \mathrm{res}_A(T) = \bigcup_{a \in A} \mathrm{res}_a(T)$.

An important feature of reaction systems is the assumption about the *non-permanency* of objects: The objects carried over to the next step are only those produced by reactions. All the other objects vanish, even if they are not involved in any reaction.

Given an initial set D_0, the semantics of a *closed* reaction system can be simply defined as the *result sequence*, $\delta =$

$D_1, \ldots, D_n$ where each set D_i, for $i \geq 1$, is obtained from the application of reactions of $\mathcal{A}$ to the state obtained at the previous step D_{i-1} ; formally, $D_i = res_{\mathcal{A}}(D_{i-1})$ for all $1 \leq i < n$. The sequence of states of the reaction system coincides with the result sequence $\delta = D_1, \ldots, D_n$. In Dennunzio et al. (2015a, b), the authors introduced the idea of *preimage* and nth *ancestor*. For simplicity, we first define them for a single product s.

Definition 1 (nth Ancestors for s) Let $\mathcal{A} = (S, A)$ be a reaction systems and $s \in S$. A set D_0 is an nth *ancestor* of s if $s \in res_{\mathcal{A}}^{(n)}(D_0)$. D_0 is called a *preimage* of s if D_0 is a 1st ancestor of s.

The same concepts can be naturally extended to sets of products.

Definition 2 (nth Ancestors for $\{s_1, \ldots, s_m\}$) Let $\mathcal{A} = (S, A)$ be a reaction systems and consider$\{s_1, \ldots, s_m\} \subseteq S$. A set D_0 is an nth *ancestor* of $\{s_1, \ldots, s_m\}$ if $\forall i : 1 \leq i \leq m, s_i \in res_{\mathcal{A}}^{(n)}(D_0)$. D_0 is called a *preimage* of $\{s_1, \ldots, s_m\}$ if D_0 is a 1st ancestor of $\{s_1, \ldots, s_m\}$.

3 Causal predicates in reaction systems

In our formulas, we use objects of reaction systems as propositional symbols. Formally, we introduce the set F_S of propositional formulas on S defined in the standard way: $S \cup \{true, false\} \subseteq F_S$ and $\neg f_1, f_1 \vee f_2, f_1 \wedge f_2 \in F_S$ if $f_1, f_2 \in F_S$. The propositional formulas F_S are interpreted with respect to C a subset of objects S. Intuitively, $s \in C$ denotes the presence of element s and therefore the truth of the corresponding propositional symbol. The complete definition of the formula satisfaction relation is as follows.

Definition 3 Let $C \subseteq S$ for a given set of objects S. Given a propositional formula $f \in F_S$, the *satisfaction relation* $C \models f$ is inductively defined as follows:

$C \models true$,
$C \models s$ iff $s \in C$,
$C \models \neg f'$ iff $C \not\models f'$,
$C \models f_1 \vee f_2$ iff either $C \models f_1$ or $C \models f_2$,
$C \models f_1 \wedge f_2$ iff $C \models f_1$ and $C \models f_2$.

In the following, $\equiv$ stands for the logical equivalence on propositional formulas F_S. Moreover, given a formula $f \in F_S$, with $atom(f)$ we denote the set of propositional symbols that appear in f.

Given a formula f, a disjunctive normal form (DNF) of f can be computed by applying the following procedure:

1. Put the negations next to the atomic objects using De Morgan's laws;
2. Put the conjunctions within the disjunctions using the distributive law;
3. Simplify the obtained formula using the idempotent, negation, domination and negation laws.

Alternatively, we can construct the complete DNF of f by constructing the truth table of f and representing with a conjunction all rows that have a truth value 1. It is worth noting that both methods are exponential in the worst case. Indeed, in the first method the application of the distributive laws (step 2) can be exponential, while in the second method the construction of the truth table is exponential in the number of variables of f.

Any DNF of the formula allows us to efficiently solve problems such as determining the existence of a preimage or of an nth ancestor (see Sect. 5 for more details) or to find the minimum-cardinality preimage or nth ancestor. However, although not strictly necessary, it can be convenient to consider a compact DNF representation of f so that it can be more easily verified. This requires a *logic minimization step* that consists in further simplifying the formula, in exact or heuristic way, in order to derive a DNF minimal with respect to the number of terms (i.e., conjunctions) or to the number of literals occurring in it or to any other given cost metric.

This last step is computationally expensive, as logic minimization is an NP-hard problem[1] (Hassoun and Sasao 2002; Umans et al. 2006). However, since near-minimum solutions are sufficient, the logic minimization step can be performed applying heuristic methods to produce solutions that are near to the optimum in a relatively short time. In particular, in our setting, we are interested in deriving a compact logic expression containing only *essential* propositional symbols, i.e., symbols on which the expression actually depends. Thus, in this context, we can apply heuristic techniques (e.g., the ESPRESSO heuristic minimizer Brayton et al. 1984) to produce near minimal *prime* and *irredundant* DNF formulas, i.e., DNF where each conjunction corresponds to a prime implicant[2] of the function represented by the expression (primality), and no conjunction can be deleted without changing the function represented by the expression (irredundancy). Indeed, nonessential propositional symbols cannot appear in any prime implicant of a given Boolean function.

Let us denote with $min(f)$ the DNF obtained after the exact or heuristic logic minimization step. For any formula $f \in F_S$, $min(f)$ is equivalent to f and is minimal with respect to the number of terms (i.e., conjunctions) or to the

[1] More precisely, the decision version of the problem of finding a minimal DNF representation of a Boolean function f starting from its truth table is NP-complete, while it becomes NP^{NP}-complete starting from a DNF for f.

[2] A prime implicant of a Boolean function f is a conjunction of literals, that implies f and s. t. removing any literal results in a new conjunction that does not imply f.

number of literals occurring in it or to any other chosen cost function. Thus, for any reasonable cost function we have $f \equiv min(f)$ and $atom(min(f)) \subseteq atom(f)$ and there exists no formula f' such that $f' \equiv f$ and $atom(f') \subset atom(min(f))$.

The causes of an object in a reaction system are defined by a propositional formula on the set of objects S. First of all, we define the *applicability predicate* of a reaction a as a propositional logic formula on S describing the requirements for applicability of a; namely, all reactants have to be present and inhibitors have to be absent. This is represented by the conjunction of all atomic formulas representing reactants and the negations of all atomic formulas representing inhibitors of the considered reaction.

Definition 4 Let $a = (R, I, P)$ be a reaction with $R, I, P \subseteq S$ for a set of objects S. The *applicability predicate* of a, denoted by $ap(a)$, is defined as follows:

$$ap(a) = \left(\bigwedge_{s_r \in R} s_r \right) \wedge \left(\bigwedge_{s_i \in I} \neg s_i \right).$$

The *causal predicate* of a given object s is a propositional formula on S representing the conditions for the production of s in one step; namely, at least one reaction having s as a product has to be applicable.

Definition 5 Let $\mathcal{A} = (S, A)$ be a reaction systems and $s \in S$. The *causal predicate* of s in $\mathcal{A}$, denoted by $cause(s, \mathcal{A})$ (or $cause(s)$, when $\mathcal{A}$ is clear from the context), is defined as follows[3]:

$$cause(s, \mathcal{A}) = \bigvee_{\{(R,I,P) \in A \mid s \in P\}} ap\left((R, I, P)\right).$$

Moreover, we define the *negation of the causal predicate* of s, called $nocause(s)$, as the DNF formula equivalent to $\neg cause(s)$.

Note that $nocause(s)$ can easily be computed (once for all) from $cause(s)$ by putting the negations next to the atomic objects using De Morgan's laws and then by putting the conjunctions within the disjunctions using the distributive law.

We now introduce a simple reaction system as running example.

Example 1 Let $\mathcal{A}_1 = (\{A, \ldots, G\}, \{a_1, a_2, a_3\})$ be a reaction system with the following reactions,

$$a_1 = (\{A\}, \emptyset, \{B\})$$
$$a_2 = (\{C, D\}, \emptyset, \{E, F\})$$
$$a_3 = (\{G\}, \{B\}, \{E\}).$$

[3] We assume that $cause(s) = false$ if there is no $(R, I, P) \in A$ such that $s \in P$.

The *applicability predicates* of the reactions are $ap(a_1) = A$, $ap(a_2) = C \wedge D$ and $ap(a_3) = G \wedge \neg B$. Thus, the *causal predicates* of the objects are

$$cause(A) = cause(C) = cause(D) = cause(G) = false,$$
$$cause(B) = A,$$
$$cause(F) = C \wedge D,$$
$$cause(E) = (C \wedge D) \vee (G \wedge \neg B),$$

while

$$\begin{aligned} nocause(A) &= nocause(C) = nocause(D) \\ &= nocause(G) = true, \\ nocause(B) &= \neg A, \\ nocause(F) &= \neg C \vee \neg D, \\ nocause(E) &= (\neg C \wedge \neg G) \vee (\neg C \wedge B) \\ &\quad \vee (\neg D \wedge \neg G) \vee (\neg D \wedge B). \end{aligned}$$

4 Characterizing the *n*th ancestors

We aim to define a formula characterizing all the initial sets D_0 that lead to the production of a given product $s \in S$ after exactly n steps. Note that the formula for the nth ancestor of a set of products $\{s_1, s_2, \ldots, s_m\} \subseteq S$ can be obtained by combining in conjunction all the nth ancestor formulas for each s_i with $i \in \{1, \ldots, m\}$ (see Corollary 1).

We base our new definitions on the notions of *formula-based* and *specialized formula predictors*, originally presented in Barbuti et al. (2016a, b, 2018c), that characterize all causes of an object in a given number of steps. Following this approach, all nth ancestors of an object s are characterized by a propositional formula f, i.e., they are all initial sets D_0 that satisfy f according to the satisfaction relation defined in Definition 3.

Definition 6 (nth Ancestor Formula) Let $\mathcal{A} = (\mathcal{S}, \mathcal{A})$ be a reaction systems, $s \in S$ and $f \in F_S$ a propositional formula. We say that formula f *is a nth ancestor formula* of s if it holds that $D_0 \models f \Leftrightarrow s \in D_n$.

Intuitively, formula f has to characterize *all* initial states leading to the production of s in exactly n steps. Since the evolution of closed reaction systems is deterministic, there cannot exist two non-equivalent formulas f and f' both satisfying Definition 6. However, if f is an nth ancestor formula of s and $f' \equiv f$ then also f' is an nth ancestor formula of s. Among all the equivalent formulas, it is convenient to choose the one containing the minimal number of propositional symbols, so that they do not contain inessential objects. This is formalized by the following approximation order on F_S.

Definition 7 (Approximation Order) Given $f_1, f_2 \in F_S$ we say that $f_1 \sqsubseteq_f f_2$ if and only if $f_1 \equiv f_2$ and $atom(f_1) \subseteq atom(f_2)$.

It can be shown that there exists a *unique equivalence class* of nth ancestor formulas of s that is minimal with respect to the order $\sqsubseteq_f$.

We now define an operator $\mathtt{Anc}$ that allows nth ancestor formulas to be effectively computed.

Definition 8 Let $\mathcal{A} = (S, A)$ be a reaction systems and $s \in S$. We define a function $\mathtt{Anc} : S \times \mathbb{N} \to F_S$ as follows: $\mathtt{Anc}(s, n) = \mathtt{Anc}_a(cause(s), n-1)$, where the auxiliary function $\mathtt{Anc}_a : F_S \times \mathbb{N} \to F_S$ is recursively defined as follows:

$$
\begin{aligned}
&\mathtt{Anc}_a(l, 0) && = l \text{ where } l = s \text{ or } l = \neg s\\
&\mathtt{Anc}_a(s, i), \ i > 0 && = \mathtt{Anc}_a(cause(s), i-1)\\
&\mathtt{Anc}_a(\neg s, i), i > 0 && = \mathtt{Anc}_a(nocause(s), i-1)\\
&\mathtt{Anc}_a(f_1 \vee f_2, i) && = \mathtt{Anc}_a(f_1, i) \vee \mathtt{Anc}_a(f_2, i)\\
&\mathtt{Anc}_a(f_1 \wedge f_2, i) && = \mathtt{Anc}_a(f_1, i) \wedge \mathtt{Anc}_a(f_2, i)\\
&\mathtt{Anc}_a(true, i) && = true\\
&\mathtt{Anc}_a(false, i) && = false.
\end{aligned}
$$

The function $\mathtt{Anc}(\_, n)$ gives an nth ancestor formula that, in general, is not in DNF form and may not be minimal with respect to $\sqsubseteq_f$.

For this purpose, we could apply heuristic techniques to produce *prime* and *irredundant* quasi-minimal DNF that are guaranteed to be minimal with respect to $\sqsubseteq_f$.

Theorem 1 *Let $\mathcal{A} = (S, A)$ be a reaction systems. For any object $s \in S$,*

- $\mathtt{Anc}(s, n)$ *is the* nth ancestor formula *of* s*;*
- $min(\mathtt{Anc}(s, n))$ *is the* nth ancestor formula *of* s *and is* minimal *with respect to* $\sqsubseteq_f$.

The proof of the previous result can be obtained by revisiting the proof of Theorem 4.4 and Corollary 4.7 in Barbuti et al. (2016a). The previous result extends naturally to sets as follows.

Corollary 1 *Let $\mathcal{A} = (S, A)$ be a reaction systems. Given a set of objects $\{s_1, \ldots, s_m\} \subseteq S$,*

- $\bigwedge_{i \in \{1,\ldots,m\}} \mathtt{Anc}(s_i, n)$ *is a* nth ancestor formula *of* $\{s_1, \ldots, s_m\}$*;*
- $min\left(\bigwedge_{i \in \{1,\ldots,m\}} \mathtt{Anc}(s_i, n)\right)$ *is a* nth ancestor formula *of* $\{s_1, \ldots, s_m\}$ *and it is* minimal *with respect to* $\sqsubseteq_f$.

These constructive (minimal) characterizations of nth ancestors can be exploited either for studying causal dependences between objects or for solving the computational problems studied in Dennunzio et al. (2015a, b). In particular, nth ancestor formulas can be exploited for studying the complexity of checking the existence of preimages and nth ancestors, and of computing minimal preimages and nth ancestors, as we will discuss in the rest of this paper.

Example 2 Let us consider again the reaction system $\mathcal{A}_1$ of Example 1. Assume we are interested in the 1st ancestor formula of E. Hence, we calculate it by applying the function $\mathtt{Anc}$:

$$
\begin{aligned}
\mathtt{Anc}(E, 1) &= \mathtt{Anc}_a\big((G \wedge \neg B) \vee (C \wedge D), 0\big)\\
&= \big(\mathtt{Anc}_a(G, 0) \wedge \neg \mathtt{Anc}_a(B, 0)\big)\\
&\quad \vee \big(\mathtt{Anc}_a(C, 0) \wedge \mathtt{Anc}_a(D, 0)\big)\\
&= (G \wedge \neg B) \vee (C \wedge D).
\end{aligned}
$$

An initial set D_0 satisfies $\mathtt{Anc}(E, 1)$ iff the execution of the reaction system starting from D_0 leads to the production of object E after 1 step. Furthermore, in this case the obtained formula is also minimal given that $min(\mathtt{Anc}(E, 1)) = \mathtt{Anc}(E, 1)$ since $\mathtt{Anc}(E, 1)$ is already in minimal DNF. The 1st ancestors (or preimages) of E are the sets D_0 satisfying $\mathtt{Anc}(E, 1)$. They are all the possible sets containing either G but not B or both C and D. In this case, no object is really *necessary* for the production of E in one step since we have two alternative ways to produce E involving different objects.

The 2nd ancestor formula of E is equal to $false$,

$$
\begin{aligned}
\mathtt{Anc}(E, 2) &= \mathtt{Anc}_a\big((G \wedge \neg B) \vee (C \wedge D), 1\big)\\
&= \big(\mathtt{Anc}_a(G, 1) \wedge \neg \mathtt{Anc}_a(B, 1)\big)\\
&\quad \vee \big(\mathtt{Anc}_a(C, 1) \wedge \mathtt{Anc}_a(D, 1)\big)\\
&= (\mathtt{Anc}_a(false, 0) \wedge \neg \mathtt{Anc}_a(A, 0))\\
&\quad \vee (\mathtt{Anc}_a(false, 0) \wedge \mathtt{Anc}_a(false, 0))\\
&= (false \wedge \neg A) \vee (false \wedge false)\\
&\equiv false.
\end{aligned}
$$

This means that there does not exist any 2nd ancestor of E, that is, no initial state can lead to E in two steps. Of course, also any nth ancestor formula of E with $n > 2$ is equal to $false$. Therefore, we can conclude that there does not exist any nth ancestor of E for any $n \geq 2$.

Example 3 Let us consider now the reaction system $\mathcal{A}_2 = (\{A, \ldots, L\}, \{a_1, \ldots, a_8\})$ with the following reaction rules:

$$
\begin{aligned}
a_1 &= (\{A\}, \{B\}, \{C\}) & a_2 &= (\{C\}, \emptyset, \{E, I\})\\
a_3 &= (\{G, B\}, \emptyset, \{D\}) & a_4 &= (\{B\}, \emptyset, \{B\})\\
a_5 &= (\{H, B\}, \emptyset, \{D\}) & a_6 &= (\{E, D\}, \emptyset, \{F\})\\
a_7 &= (\{I\}, \emptyset, \{G\}) & a_8 &= (\{L\}, \emptyset, \{H\}).
\end{aligned}
$$

Assume we are interested in the preimages of F. We obtain the 1st ancestor formula $\mathtt{Anc}(F, 1) = E \wedge D$, expressing

that any set containing $\{E, D\}$ is preimage of F. Looking for the 2nd ancestors of F, we obtain

$$\begin{aligned}\mathtt{Anc}(F,2) &= \mathtt{Anc}_a\big((E \wedge D), 1\big)\\ &= \big(\mathtt{Anc}_a(E,1) \wedge \mathtt{Anc}_a(D,1)\big)\\ &= \mathtt{Anc}_a(C,0)\\ &\quad \wedge \mathtt{Anc}_a((G \wedge B) \vee (H \wedge B), 0)\\ &= C \wedge ((\mathtt{Anc}_a(G,0) \wedge \mathtt{Anc}_a(B,0))\\ &\quad \vee (\mathtt{Anc}_a(H,0) \wedge \mathtt{Anc}_a(B,0)))\\ &= C \wedge ((G \wedge B) \vee (H \wedge B)).\end{aligned}$$

Note that $min(\mathtt{Anc}(F,2)) = (C \wedge G \wedge B) \vee (C \wedge H \wedge B)$ in this case is simply obtained by applying the distributive law that gives an already minimized DNF.

The 2nd ancestor formula expresses that any set containing either $\{C, G, B\}$ or $\{C, H, B\}$ is a 2nd ancestors of F. Note that we can conclude that while C and B are *necessary* for the production of F in two steps, G and H are not.

As regards the 3rd ancestors, we have

$$\begin{aligned}\mathtt{Anc}(F,3) &= \mathtt{Anc}_a\big((E \wedge D), 2\big)\\ &= \big(\mathtt{Anc}_a(E,2) \wedge \mathtt{Anc}_a(D,2)\big)\\ &= \mathtt{Anc}_a(C,1)\\ &\quad \wedge \mathtt{Anc}_a((G \wedge B) \vee (H \wedge B)), 1)\\ &= \mathtt{Anc}_a(C,1)\\ &\quad \wedge ((\mathtt{Anc}_a(G,1) \wedge \mathtt{Anc}_a(B,1)\\ &\quad \vee (\mathtt{Anc}_a(H,1) \wedge \mathtt{Anc}_a(B,1)))\\ &= \mathtt{Anc}_a(A \wedge \neg B, 0)\\ &\quad \wedge (\mathtt{Anc}_a(I,0) \wedge \mathtt{Anc}_a(B,0)\\ &\quad \vee (\mathtt{Anc}_a(L,0) \wedge \mathtt{Anc}_a(B,0)))\\ &= A \wedge \neg B \wedge ((I \wedge B) \vee (L \wedge B)).\end{aligned}$$

This time $min(\mathtt{Anc}(F,3)) = false$; therefore, we can be sure that it does not exist any nth ancestor of F for any $n > 2$.

The concept of ancestor is very useful for reasoning on reaction systems modeling gene regulatory networks. A gene regulatory network can be easily translated into a reaction system (Barbuti et al. 2018b). The dynamics of the reaction system, then, precisely describes the evolution of the gene regulatory network. In this context, the nth ancestors describe initial configurations of active/inactive genes that lead to a given final configuration after n steps.

Example 4 Consider the well-known *repressilator* network (Elowitz and Leibler 2000) depicted in Fig. 1. It consists of three genes connected in a feedback loop, where each gene inhibits the expression of the next gene in the loop.

The repressilator network can be naturally expressed as a the reaction system
$\mathcal{A}_{Repressillator} = (\{X, Y, Z\}, \{a_1, a_2, a_3\})$ with the following reaction rules:

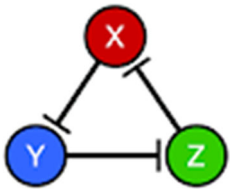

Fig. 1 Repressilator

$$\begin{aligned}a_1 &= (\emptyset, \{Z\}, \{X\})\\ a_2 &= (\emptyset, \{X\}, \{Y\})\\ a_3 &= (\emptyset, \{Y\}, \{Z\}).\end{aligned}$$

This reaction system behaves differently depending on whether the initial configuration is, for example, $\{X, Y, Z\}$ or simply $\{X\}$. However, in all cases the behavior is oscillatory. This can be shown by computing the 3rd and 6th ancestors of Z that turn out to be

$$\begin{aligned}\mathtt{Anc}(Z,3) &= \mathtt{Anc}_a(\neg Y, 2) = \mathtt{Anc}_a(X,1)\\ &= \mathtt{Anc}_a(\neg Z, 0) = \neg Z\end{aligned}$$

and

$$\begin{aligned}\mathtt{Anc}(Z,6) &= \mathtt{Anc}_a(\neg Y, 5) = \mathtt{Anc}_a(X,4) = \mathtt{Anc}_a(\neg Z, 3)\\ &= \mathtt{Anc}_a(Y,2) = \mathtt{Anc}_a(\neg X, 1) = \mathtt{Anc}_a(Z,0)\\ &= Z.\end{aligned}$$

This shows that independently of the initial configuration, the gene Z repeatedly appears and disappears, exhibiting an oscillatory behavior.

5 Complexity of ancestor problems

In this section, we investigate the complexity of solving ancestor problems using the new characterization computed by the Anc operator. In order to achieve this goal, we first study the

structure of the formula result of the Anc operator. The idea is to give an upper bound both to the size and to the depth of the formula, where depth means the number of nested AND/OR levels.

To this aim, given a reaction system $\mathcal{A} = (S, A)$, we first define some auxiliary notions. To give an upper bound to the size of the obtained formula, we need to know: (i) the maximum number of reactants and inhibitors in a rule of the system, denoted by $cp(A)$, and (ii) the maximum number of rules that produce the same object, denoted $mp(A)$.

Definition 9 Given a reaction system $\mathcal{A} = (S, A)$, let

$$cp(A) = max\{|R| + |I| \mid (R, I, P) \in A\},$$
$$mp(A) = max\{|\{(R, I, P) \in A \mid s \in P\}| \mid s \in S\},$$

where $|X|$ indicates the cardinality of the set X.

First observe that for each $s \in S$ the size of the formula $cause(s)$ in terms of number of literals is at most $cp(A) \times mp(A)$. Computing the nth ancestor formula requires n steps. At the first step, the formula computed by our operator is $cause(s)$, for some s, whose maximal size is at most $cp(A) \times mp(A)$. At the second step, each one of the $cp(A) \times mp(A)$ literals of the previous formula has to be substituted with its causes, obtaining a new formula whose size is at most $(cp(A) \times mp(A))^2$ and so on. Hence, the size of the resulting formula $\texttt{Anc}(s, n)$ is at most $(cp(A) \times mp(A))^n$ for each $s \in S$; namely, the size of $\texttt{Anc}(s, n)$ is polynomial in $cp(A)$ and $mp(A)$, as long as n has a constant value. Therefore, the size of the nth ancestor formula for the set $\{s_1, s_2, \ldots, s_m\} \subseteq S$ is $m \times (cp(A) \times mp(A))^n$ which is polynomial in m, $cp(A)$ and $mp(A)$ as long as n has a constant value.

Let us now evaluate the depth of the formula. To this aim, the idea is to measure the level of nesting of $\wedge$-$\vee$ operators. Intuitively, formula A is level 0, formulas $A \wedge B \wedge C$, $A \wedge (B \wedge C)$ and $A \vee B$ are level 1, formulas $A \wedge (B \wedge C) \vee (C \wedge D)$ and $A \vee (B \wedge C)$ are level 2, and so on.

Definition 10 Let $f \in F_S$, we call *nesting level* of f the maximum depth of its representation through a AND–OR tree.

In order to bound the nesting level of $\texttt{Anc}(s, n)$, we define the following:

Definition 11 Given a reaction system $\mathcal{A} = (S, A)$, we define

$$c(A) = \begin{cases} 1 & \text{if } cp(A) > 1; \\ 0 & \text{otherwise.} \end{cases} \qquad p(A) = \begin{cases} 1 & \text{if } mp(A) > 1; \\ 0 & \text{otherwise.} \end{cases}$$

The next result bounds the nesting level of the formula $\texttt{Anc}(s, n)$, for $s \in S$.

Theorem 2 *Let $\mathcal{A} = (S, A)$ be a reaction system. For each $s \in S$, the nesting level of the formula $\texttt{Anc}(s, n)$, characterizing the nth ancestors of s, is at most $n \times (c(A) + p(A))$.*

Proof Follows immediately from the definitions of Anc and of $cause$ and $nocause$. □

For a set $\{s_1, s_2, \ldots, s_m\} \subseteq S$, we have the following result.

Corollary 2 *Let $\mathcal{A} = (S, A)$ be a reaction system. The nesting level of the formula $\bigwedge_{i \in \{1,\ldots,m\}} \texttt{Anc}(s_i, n)$, characterizing the nth ancestors of $\{s_1, s_2, \ldots, s_m\}$, is at most $1 + n \times (c(A) + p(A))$.*

We can then conclude that for a set $\{s_1, s_2, \ldots, s_m\} \subseteq S$ our operator computes a $1 + n \times (c(A) + p(A))$ depth formula with at most $m \times (cp(A) \times mp(A))^n$ literals in polynomial time.

The formula that we obtain with our operator completely characterizes all nth ancestors, but it can be very complex to read because of the nesting levels of AND–OR operators. While checking whether a given set D_0 is an nth ancestor can be done in linear time in the size of the formula obtained by Anc, finding whether an nth ancestor exists or compute one of the minimal sizes requires some further manipulation of the formula. This is the expensive step, since putting the formula in DNF is in general exponential in the size of the formula. However, it is worth noting that once the formula is in DNF both problems of establishing whether an nth ancestor of a given product exists, and finding an nth ancestor with a minimal number of objects can be solved in polynomial time. Indeed, the satisfiability of a formula in DNF can be checked in a time that is linear in the size of the formula[4]; analogously, an ancestor of minimal size can be found in linear time scanning the DNF formula in order to select the conjunction with the minimal number of positive literals.

Therefore, we can conclude that the expensive step is to put the formula in DNF. Whenever this is not necessary because the formula resulting from our operator is already in DNF, then the complexity of the ancestor problems boils down to be polynomial.

This is the case of preimages of single objects as stated by the following result.

Theorem 3 *Let $\mathcal{A} = (S, A)$ be a reaction system. For an object $s \in S$, to establish if there exists a preimage and finding one of the minimal sizes can be done in polynomial time.*

Proof The formula $\texttt{Anc}(s, 1)$ is, by definition, in DNF. □

This is not, in general, the case for preimages of $\{s_1, \ldots s_m\}$ with $m > 1$ and for nth ancestors where $n > 1$. To handle efficiently these problems, our investigation proceeds in two different directions. On the one hand (see Sect. 6), we define conditions on the reaction system rules so that our operator computes a formula that is already in DNF. On the other hand (see Sect. 7), we explore approximations of the nth ancestor formula that can be computed in polynomial time and are useful to address ancestor problems.

[4] A formula in DNF is satisfiable if and only if at least one of its conjunctions is satisfiable; a conjunction is satisfiable if and only if it does not contain both a symbol x and its complement $\neg x$.

6 Synctactic restrictions on reactions systems

We now investigate syntactic conditions under which the existence and the minimal size of nth ancestor, with $n > 1$, can be computed in polynomial time. The first condition we introduce is *linear dependency*.

Definition 12 Let $\mathcal{A} = (S, A)$ be a reaction systems. The *n-linear dependency* of an object s_2 from an object s_1, denoted $s_1 \overset{n}{\hookrightarrow} s_2$, is recursively defined as follows:

1. $s_1 \overset{0}{\hookrightarrow} s_1$
2. $s_1 \overset{1}{\hookrightarrow} s_2$ iff $|\{(R, I, P) \mid (R, I, P) \in A \text{ and } s_2 \in P\}| = 1$ and $|R| + |I| = 1$;
3. $s_1 \overset{n}{\hookrightarrow} s_2$ with $n > 1$ iff there exists $s_3 \in S$ such that $s_1 \overset{1}{\hookrightarrow} s_3$ and $s_3 \overset{n-1}{\hookrightarrow} s_2$.

Intuitively, s_2 is n-linearly dependent from s_1 if it can be produced from s_1 in n steps in a unique way and by producing a single element at each step. Moreover, we say that s_2 is n-linearly produced if there exists a s_1 such that either s_2 is n-linearly dependent from s_1 or s_2 is n'-linearly dependent from s_1 with $n' < n$ and s_1 does not appear as product of any reaction. Formally,

Definition 13 Let $\mathcal{A} = (S, A)$ be a reaction system. An object s_2 is *n-linearly produced* in $\mathcal{A}$ iff one of the following conditions holds:

1. there exists s_1 such that $s_1 \overset{n}{\hookrightarrow} s_2$ or
2. there exists $0 \leq n' < n, s_1 \in S$ such that $s_1 \overset{n'}{\hookrightarrow} s_2$ and $|\{(R, I, P) \mid (R, I, P) \in A \text{ and } s_1 \in P\}| = 0$.

We are now ready to prove that for reaction systems that have at most one rule producing each object and such that all inhibitors are n-linearly produced, the nth ancestor problems are much easier to be solved.

Theorem 4 *Let $\mathcal{A} = (S, A)$ be a reaction system satisfying the following conditions:*

1. *$\forall s \in S$, $|\{(R, I, P) \mid (R, I, P) \in A \text{ and } s \in P\}| \leq 1$ and*
2. *for every object $s_i \in I$ with $(R, I, P) \in A$, s_i is n-linearly produced.*

Then, for any $s \in S$, the existence and minimal size of the nth ancestors of s can be solved in polynomial time.

Proof In order to prove the claim, we prove that in this case the operator `Anc` of Definition 8 gives a formula which is already in DNF.

Since for every object $s \in S$ Condition 1 is satisfied, we know that for each $s \in S$ the formula $causes(s)$ is a conjunction of literals. Moreover, every object used as inhibitor is n-linearly produced; thus, every negated object $\neg s'$, in the computation of n steps, is replaced by $nocause(s')$ which consists of a single literal. As a consequence the disjunction operator is never introduced. Hence, the nth ancestor formula of s will be simply a conjunction of literals (objects or non-negated objects) of nesting level 1 which is a particular case of DNF. □

The next result extends the previous one to sets of objects.

Corollary 3 *Let $\mathcal{A} = (S, A)$ be a reaction system. If all the conditions of Theorem 4 are satisfied, then, for any set $\{s_1, s_2, \ldots, s_m\} \subseteq S$, the existence and minimal size of the nth ancestors can be solved in polynomial time.*

Proof As before, also in this case, we prove that the operator `Anc` of Definition 8 gives a formula which is already in DNF. The nth ancestor formula of a set of objects is, by definition, a conjunctive formula of nesting level 1. Hence, the whole nth ancestor formula of $\{s_1, s_2, \ldots, s_m\}$ is still a conjunctive formula of nesting level 1. □

Example 5 Let $\mathcal{A}_3 = (\{A, \ldots, G\}, \{a_1, \ldots, a_6\})$ be a reaction system with

$$a_1 = (\{A, B\}, \emptyset, \{C\}) \qquad a_2 = (\{D, E\}, \{F\}, \{A\})$$
$$a_3 = (\{G\}, \emptyset, \{F\}) \qquad a_4 = (\{B\}, \emptyset, \{G\})$$
$$a_5 = (\{C, B\}, \emptyset, \{D\}) \qquad a_6 = (\{E\}, \emptyset, \{E\})$$
$$a_7 = (\{B\}, \emptyset, \{B\}).$$

Since every object is produced by at most one rule, we have that condition 1 of Theorem 4 is satisfied. Moreover, the inhibitors B and F are (linearly) produced by a reaction with a single reactant which, in turn, is (linearly) produced by a reaction with a single reactant; then, also the second condition of Theorem 4 and Corollary 3 is satisfied for all n. Now $\texttt{Anc}(A, 2) = C \wedge B \wedge E \wedge \neg G$, while $\texttt{Anc}(A, 3) = A \wedge B \wedge B = A \wedge B$ and $\texttt{Anc}(A, 4) = false$. Note that as stated in the proof of Theorem 4, all ancestors computed by `Anc` for the reaction system $\mathcal{A}_3$ are formulas already in DNF.

Consider now a reaction system where reaction a_4 is substituted by reaction $a_4' = (\{B\}, \emptyset, \{G\})$. In this case, $\texttt{Anc}(A, 2) = C \wedge B \wedge E \wedge \neg G$, but $\texttt{Anc}(A, 3) = A \wedge B \wedge E \wedge \neg B = false$.

Conditions expressed by Theorem 4 and Corollary 3 are not a characterization of all reaction systems having an nth ancestor formula in DNF. Weaker conditions can be found as, for instance, in the following proposition. The idea in this case

is to allow one single object to be produced by more than one reaction.

Proposition 1 *et $\mathcal{A} = (S, A)$ be a reaction system satisfying the following conditions:*

1. *there exists $\overline{s} \in S$ such that $|\{(R, I, P) \mid (R, I, P) \in A \text{ and } \overline{s} \in P\}| > 1$ and*
2. *$\forall s \in S \backslash \{\overline{s}\}$. $|\{(R, I, P) \mid (R, I, P) \in A \text{ and } s \in P\}| \leq 1$ and*
3. *for every object $s_i \in I$ with $(R, I, P) \in A$, s_i is n-linearly produced.*

Then, for any $s \in S$, the existence and minimal size of the nth ancestors of s can be solved in polynomial time.

Proof Even in this case, we prove that the operator `Anc` of Definition 8 gives a formula which is already in DNF. Since there is only $\overline{s} \in S$ that is produced by more than one rule, only $cause(\overline{s})$ can contain the OR operator. Moreover, $\overline{s}$ cannot appear as reactant in any rule. Thus, it cannot be introduced in the computation of the nth ancestor formula of any other object. Further, every object used as inhibitor is n-linearly produced; thus, every negated object s', in the computation of n steps, is replaced by $nocause(s')$ which is a single literal. Then, the nth ancestor formula of s is a DNF. □

6.1 A biological application: the lac operon

We consider the regulation process of the lactose operon in *E. coli* (*Escherichia coli*, see Wong et al. 1997; Vilar et al. 2003). *E. coli* is a bacterium often present in the intestine of many animals. As most bacteria, it is often exposed to a constantly changing physical and chemical environment, and reacts to changes in its environment through changes in the kinds of proteins it produces.

In general, in order to save energy, bacteria do not synthesize degradative enzymes unless the substrates for these enzymes are present in the environment. For example, *E. coli* does not synthesize the enzymes that degrade lactose unless lactose is in the environment. This phenomenon is called *enzyme induction* or, more generally, *gene regulation* since it is obtained by controlling the transcription of some genes into the corresponding proteins.

Let us consider the lactose degradation example in *E. coli*. Two enzymes are required to start the breaking process: the *lactose permease*, which is incorporated in the membrane of the bacterium and actively transports the sugar into the cell (without this enzyme lactose can enter the bacterium anyway, but much more slowly), and the *beta galactosidase*, which splits lactose into glucose and galactose. The bacterium produces also the *transacetylase* enzyme, whose function is marginal.

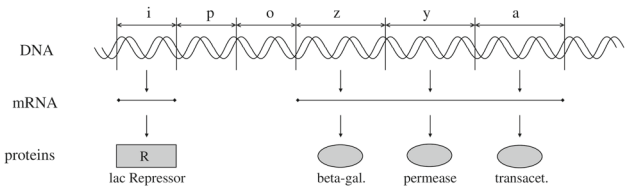

Fig. 2 Lac operon

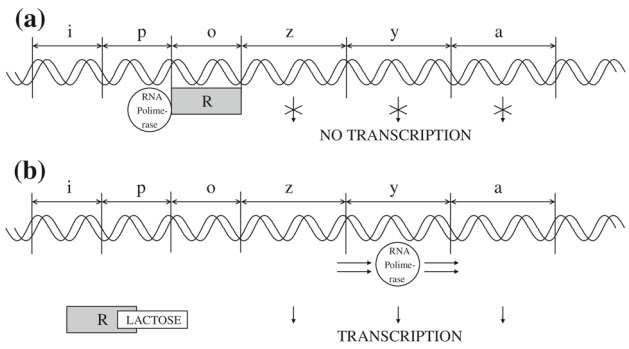

Fig. 3 Regulation process

The sequence of genes in the DNA of *E. coli* which produces the described enzymes is known as the *lactose operon* (see Fig. 2). It is composed by six genes: The first three (i, p, o) regulate the production of the enzymes, and the last three (z, y, a), called *structural genes*, are transcribed (when allowed) into the mRNA for beta galactosidase, lactose permease and transacetylase, respectively.

The regulation process is as follows (see Fig. 3): Gene i encodes the *lac Repressor*, which, in the absence of lactose, binds to gene o (the *operator*). Transcription of structural genes into mRNA is performed by the RNA polymerase enzyme, which usually binds to gene p (the *promoter*) and scans the operon from left to right by transcribing the three structural genes z, y and a into a single mRNA fragment. When the lac Repressor is bound to gene o, it becomes an obstacle for the RNA polymerase, and transcription of the structural genes is not performed. On the other hand, when lactose is present inside the bacterium, it binds to the Repressor and this cannot stop any more the activity of the RNA polymerase. In this case, transcription is performed and the three enzymes for lactose degradation are synthesized.

At a high level of abstraction, the previous complex mechanism can be described by considering only the influences that each gene has on the other genes. This is the usual approach in gene regulatory networks, which focus on positive and negative influences (promotion and inhibition) between pairs of genes, by abstracting away the biochemical details of the interaction. We follow this approach now in order to define a reaction system modeling the lac operon.

In the lac operon, the repressor R has a negative influence on the transcription and synthesis of the three enzymes for

lactose degradation. Since such three enzymes are always synthesized together, we describe them with the single symbol Z in the reaction system model. The repressor R is in turn subject to the negative influence of lactose (denoted $LACT$). In addition to these influences, we model the activity of the RNA polymerase (denoted POL), which can bind to the promoter gene p. We denote the result of such binding as Pp. As a result, we obtain the following simple reaction system: $\mathcal{A}_{Operon} = (\{LACT, p, POL, Pp, R, Z\}, \{a_1, \ldots, a_6\})$

$a_1 = (\{p\}, \emptyset\{p\})$
$a_2 = (\{POL\}, \emptyset, \{POL\})$
$a_3 = (\{LACT\}, \emptyset, \{LACT\})$
$a_4 = (\{POL, p\}, \emptyset, \{Pp\})$
$a_5 = (\{Pp\}, \{R\}, \{Z\})$
$a_6 = (\emptyset, \{LACT\}, \{R\})$.

Rules a_1, a_2 and a_3 describe the persistence of the promoter gene, of the RNA polymerase and of lactose. Rules a_4 and a_5 describe the activity of the RNA polymerase, namely the binding with the promoter gene and the transcription of the structural genes. The latter rule has R as inhibitor in order to model the negative influence of the repressor on the transcription and synthesis of the three enzymes denoted Z. Finally, rule a_6 models the negative influence of lactose on the repressor.

In this example, we are interested in the production of the three enzymes for lactose degradation, denoted Z. In particular, we are interested in knowing whether a 2nd (or higher) ancestor exists, which is the minimal set of genes that has to be active for the production of Z. This would tell us what are the minimal conditions leading to the production of the three enzymes. Since the reaction system $\mathcal{A}_{Operon}$ satisfies Theorem 4, we can be sure that the computation of the DNF of the ancestor formula will take polynomial time. For $n \geq 2$, we obtain $\texttt{Anc}(Z, n) = p \wedge POL \wedge LACT$, which tells us that $LACT$ is necessary in order to produce Z.

7 Approximations of the *n*th ancestor formula

In the previous section, we have established some syntactic restrictions that guarantee us that the formula result of our Anc operator is already in DNF. This allowed us to address, for those cases, nth ancestor problems in polynomial time. However, the syntactic restrictions of Theorem 4 are quite strong, and therefore, they allow to address only a small set of reaction systems. An alternative way to keep the complexity low is to consider approximations of the nth ancestor formula. In particular, given a Boolean formula f which is the nth ancestor formula of an object s, we want to effectively compute two DNF formulas $f^{\otimes}$ and $f^{\oplus}$ such that the following logical implications are satisfied

$$f^{\otimes} \rightarrow f \rightarrow f^{\oplus}. \tag{1}$$

It is worth noting that the properties in (1) guarantee that any initial set D_0 satisfying a $f^{\otimes}$ formula is an nth ancestor of s. Therefore, if a formula

$f^{\otimes}$ is satisfiable[5], then there exists at least an nth ancestor of s. Moreover, scanning a $f^{\otimes}$ formula and selecting the conjunction with the minimal number of positive literals gives an upper bound to the minimal size of an nth ancestor of s. On the other hand, if the formula $f^{\oplus} \equiv \mathit{false}$, then we can be sure that there are no nth ancestors of s. Moreover, if an initial D_0 does not satisfy the formula $f^{\oplus}$, then we can be sure that D_0 cannot be an nth ancestor of s.

Both kinds of formulas can be computed by effective operators in polynomial time. Such operators can be obtained as different approximations (in the sense of abstract interpretation Cousot and Cousot 1977, 1979) of the original Anc operator. In the next sections, we will show how to compute the formulas $f^{\otimes}$ and $f^{\oplus}$. The idea is then to use both kinds of approximation for reasoning on nth ancestor problems.

7.1 The approximation $f^{\otimes}$

Our aim is to compute DNF formulas implying the nth ancestor formula for an object s using approximations of the Anc(s, n) operator. Since we require that the approximations $f^{\otimes}$ imply the nth ancestor formula, it follows that each $f^{\otimes}$ formula we obtain will be satisfiable only by a subset of the nth ancestors of s. Roughly speaking, in order to obtain such formulas, the idea is to rewrite each positive literal using the conjunction of literals from a single reaction that produces s.

By changing the combinations of the chosen conjunctions, we obtain different $f^{\otimes}$ formulas. Indeed, the idea is to compute $f^{\otimes}$ formulas defining a new operator that mimics the operator Anc where at each rewriting step, each positive literal is replaced by just one possible conjunction of the DNF *cause*, while each negative literal is replaced with just one conjunction of the DNF of *nocause*. It is worth noting that this corresponds to exploring the derivation tree for s in n steps with a depth first search rather than a breath first search.

Formally, since there are many different $f^{\otimes}$ approximations of the nth ancestor formula, one for each possible conjunction choice, we define the operator computing $f^{\otimes}$ by means of a relation $\texttt{Anc}^{\otimes}$ as follows.

Definition 14 Let $\mathcal{A} = (S, A)$ be a reaction systems and $s \in S$. We define a relation $\texttt{Anc}^{\otimes} : S \times \mathbb{N} \times F_S$ where

[5] Remember that this check for a formula in DNF can be done in linear time.

$\mathtt{Anc}^{\otimes}(s, n, f^{\otimes})$ if $\mathtt{Anc}_a^{\otimes}(cause(s), n-1, f^{\otimes})$, where the auxiliary relation $\mathtt{Anc}_a^{\otimes} : F_S \times \mathbb{N} \times F_S$ is recursively defined as follows,

$$
\begin{array}{lll}
\mathtt{Anc}_a^{\otimes}(l, 0, l), & \text{if} & l = s \text{ or } l = \neg s \\
\mathtt{Anc}_a^{\otimes}(s, i, f^{\otimes}), \ i > 0 & \text{if} & \mathtt{Anc}_a^{\otimes}(t, i-1, f^{\otimes}), \\
 & & t \in ChooseC(cause(s)) \\
\mathtt{Anc}_a^{\otimes}(\neg s, i, f^{\otimes}), i > 0 & \text{if} & \mathtt{Anc}_a^{\otimes}(t, i-1, f^{\otimes}), \\
 & & t \in ChooseC(nocause(s)) \\
\mathtt{Anc}_a^{\otimes}(f_1 \vee f_2, i, f_1^{\otimes} \vee f_2^{\otimes}) & \text{if} & \mathtt{Anc}_a^{\otimes}(f_1, i, f_1^{\otimes}) \\
 & \text{and} & \mathtt{Anc}_a^{\otimes}(f_2, i, f_2^{\otimes}) \\
\mathtt{Anc}_a^{\otimes}(f_1 \wedge f_2, i, f_1^{\otimes} \wedge f_2^{\otimes}) & \text{if} & \mathtt{Anc}_a^{\otimes}(f_1, i, f_1^{\otimes}) \\
 & \text{and} & \mathtt{Anc}_a^{\otimes}(f_2, i, f_2^{\otimes}) \\
\mathtt{Anc}_a^{\otimes}(true, i, true) & & \\
\mathtt{Anc}_a^{\otimes}(false, i, false) & &
\end{array}
$$

where $ChooseC(f) = \{C_1, C_2, \ldots, C_r\}$ if $f = \bigvee_{i \in \{1,\ldots,r\}} C_i$, and $\forall i \in \{1 \ldots r\}$, C_i is a conjunction of literals.

All formulas $f^{\otimes}$ in relation $\mathtt{Anc}^{\otimes}$ with s and n approximate the nth ancestor formula of s in the sense of Eq. (1). It is worth noting that, given an object of interest s, for any n, each $f^{\otimes}$ formula such that $\mathtt{Anc}^{\otimes}(s, n, f^{\otimes})$ is a DNF formula that implies $\mathtt{Anc}(s, n)$.

Theorem 5 *Let $\mathcal{A} = (S, A)$ be a reaction systems. Let $f^{\otimes}$ such that $\mathtt{Anc}^{\otimes}(s, n, f^{\otimes})$ and $f = \mathtt{Anc}(s, n)$, for any object $s \in S$. Then*

- *$f^{\otimes}$ is a formula in disjunctive normal form,*
- *$f^{\otimes} \rightarrow f$.*

Proof Let us prove first that, if a generic formula f is a conjunction of literals, where, we recall, a literal is either an object or a negated object, and $\mathtt{Anc}^{\otimes}(f, n, f^{\otimes})$ holds, then $f^{\otimes}$ is a conjunction of literals.

We prove this assert by induction on the pair of the first two arguments of the relation $\mathtt{Anc}_a^{\otimes}$. The induction is based on the following well-founded ordering. We assume that a pair (f, n) is less than a pair (f', n'), denoted $(f, n) \prec (f', n')$ iff either $n < n'$ or $n = n'$ and f is a subformula of f'.

The minimal elements of $\prec$ are the pairs $(l, 0)$ where l is a literal. Thus, because we know that $\mathtt{Anc}_a^{\otimes}(l, 0, l)$, when $l = s$ or $l = \neg s$, the base case is obvious.

For the inductive cases, we have:

1. If $\mathtt{Anc}_a^{\otimes}(f_1, i, f_1^{\otimes})$ and $\mathtt{Anc}_a^{\otimes}(f_2, i, f_2^{\otimes})$ have the property that f_1, f_2, $f_1^{\otimes}$ and $f_2^{\otimes}$ are conjunctions of literals, then, by the definition of $\mathtt{Anc}^{\otimes}$ we have $\mathtt{Anc}_a^{\otimes}(f_1 \wedge f_2, i, f_1^{\otimes} \wedge f_2^{\otimes})$ where both $f_1 \wedge f_2$ and $f_1^{\otimes} \wedge f_2^{\otimes}$ are conjunctions of literals.
2. If $\mathtt{Anc}_a^{\otimes}(t, i, f^{\otimes})$ where $t \in ChooseC(cause(s))$, for an object s, we know that t is a conjunction of literals. Suppose $f^{\otimes}$ is a conjunction of literals, then, by the definition of relation $\mathtt{Anc}_a^{\otimes}$, we have $\mathtt{Anc}_a^{\otimes}(s, i+1, f^{\otimes})$. Because s is a single object, the property is still true.
3. The case in which we have $\mathtt{Anc}_a^{\otimes}(t, i, f^{\otimes})$ where $t \in ChooseC(nocause(s))$ is analogous to the previous one.

Note that the property is trivially preserved by the two cases $\mathtt{Anc}_a^{\otimes}(true, i, true)$ and $\mathtt{Anc}_a^{\otimes}(false, i, false)$. Let us remark that, for our assumption, we never apply the case $\mathtt{Anc}_a^{\otimes}(f_1 \vee f_2, i, f_1^{\otimes} \vee f_2^{\otimes})$ because we start with a formula which is a conjunction of literals and the operator $\vee$ is never introduced.

Finally, to prove that $f^{\otimes}$ is in DNF, we notice that $\mathtt{Anc}^{\otimes}(s, n, f^{\otimes})$ if $\mathtt{Anc}_a^{\otimes}(cause(s), n-1, f^{\otimes})$. Because $cause(s)$ is in DNF, $cause(s) = \bigvee_{i \in \{1,\ldots m\}} C_i$, where each C_i is a conjunction of literals. By definition, $\mathtt{Anc}_a^{\otimes}(\bigvee_{i \in \{1,\ldots m\}} C_i, n-1, f^{\otimes} = \bigvee_{i \in \{1,\ldots m\}} f_i^{\otimes})$ if $\mathtt{Anc}_a^{\otimes}(C_i, n-1, f_i^{\otimes})$ for $i \in \{1, \ldots, m\}$. Since C_i is a conjunction of literals, we can apply the above result to conclude that each $f_i^{\otimes}$ is a conjunction of literals. This allows us to conclude that $f^{\otimes}$ is in disjunctive normal form.

The proof of $f^{\otimes} \rightarrow f$ is trivial. If in a formula f we substitute a disjunction of subformulas with just one subformula in the disjunction, we obtain a stronger formula which implies the original one, and this is what this approximation does by substituting a disjunction of conjunctions of literals in $cause(s)$ or in $nocause(s)$ with just one conjunction of literals. □

Example 6 Let us consider the simple reaction system $\mathcal{A}_2$ of Example 3.

One approximation $f^{\otimes}$ of the 2nd ancestors of F is $C \wedge (H \wedge B)$ and it can be computed as follows.

$$
\begin{array}{lll}
\mathtt{Anc}^{\otimes}(F, 2, f^{\otimes}) & \text{if} & \mathtt{Anc}_a^{\otimes}\big((E \wedge D), 1, f^{\otimes} = f_1^{\otimes} \wedge f_2^{\otimes}\big) \\
 & \text{if} & \mathtt{Anc}_a^{\otimes}(E, 1, f_1^{\otimes}) \text{ and } \mathtt{Anc}_a^{\otimes}(D, 1, f_2^{\otimes}) \\
 & \text{if} & \mathtt{Anc}_a^{\otimes}(C, 0, f_1^{\otimes} = C) \\
 & & \text{and } \mathtt{Anc}_a^{\otimes}((H \wedge B), 0, f_2^{\otimes} = f_3^{\otimes} \wedge f_4^{\otimes}) \\
 & \text{if} & \mathtt{Anc}_a^{\otimes}(H, 0, f_3^{\otimes} = H) \\
 & & \text{and } \mathtt{Anc}_a^{\otimes}(B, 0, f_4^{\otimes} = B)
\end{array}
$$

Note that in $\mathtt{Anc}^{\otimes}(F, 2, f^{\otimes})$, $f^{\otimes} = C \wedge (H \wedge B)$ is already in DNF and $C \wedge (H \wedge B) \rightarrow \mathtt{Anc}(F, 2)$, since $\mathtt{Anc}(F, 2) = C \wedge ((G \wedge B) \vee (H \wedge B))$.

The approximation of the 2nd ancestor formula that we computed assures us that any set containing $\{C, H, B\}$ is a 2nd ancestors of F. So, if one is interested in finding a 2nd ancestor, one can obtain one in polynomial time by just rewriting a single branch of the tree.

Of course, also $C \wedge (G \wedge B)$ can be computed for the relation $\mathtt{Anc}^{\otimes}(F, 2, f^{\otimes})$ by choosing the other possible conjunction in the DNF $causes(D)$ formula.

This is not the case for the 3rd ancestor formula of F. Indeed, for the $f^{\otimes}$ approximations of the 3rd ancestor formula for F we can derive the formula $A \wedge \neg B \wedge (I \wedge B)$ as

follows

$$
\begin{array}{ll}
\mathtt{Anc}^{\otimes}(F,3,f^{\otimes}) & \text{if } \mathtt{Anc}^{\otimes}_{a}\big((E\wedge D),2,f^{\otimes}=f^{\otimes}_{1}\wedge f^{\otimes}_{2}\big)\\
& \text{if } \mathtt{Anc}^{\otimes}_{a}(E,2,f^{\otimes}_{1}) \text{ and } \mathtt{Anc}^{\otimes}_{a}(D,2,f^{\otimes}_{2})\\
& \text{if } \mathtt{Anc}^{\otimes}_{a}(C,1,f^{\otimes}_{1})\\
& \quad \text{and } \mathtt{Anc}^{\otimes}_{a}((G\wedge B),1,f^{\otimes}_{2}=f^{\otimes}_{3}\wedge f^{\otimes}_{4})\\
& \text{if } \mathtt{Anc}^{\otimes}_{a}(A\wedge\neg B,0,f^{\otimes}_{1}=f^{\otimes}_{5}\wedge f^{\otimes}_{6})\\
& \quad \text{and } \mathtt{Anc}^{\otimes}_{a}(G,1,f^{\otimes}_{3}),\mathtt{Anc}^{\otimes}_{a}(B,1,f^{\otimes}_{4})\\
& \text{if } \mathtt{Anc}^{\otimes}_{a}(A,0,f^{\otimes}_{5}=A),\mathtt{Anc}^{\otimes}_{a}(\neg B,0,f^{\otimes}_{6}=\neg B)\\
& \quad \text{and } \mathtt{Anc}^{\otimes}_{a}(I,0,f^{\otimes}_{3}=I),\mathtt{Anc}^{\otimes}_{a}(B,0,f^{\otimes}_{4}=B)
\end{array}
$$

The approximation formula is $f^{\otimes} \equiv false$, it is in DNF and implies $\mathtt{Anc}(F,3) = A\wedge\neg B\wedge((I\wedge B)\vee(L\wedge B)) \equiv false$ as stated by Theorem 5. Note, however, that in general $f^{\otimes} \equiv false$ does not allow us to conclude that there exist no 3rd ancestors of F.

7.2 The approximation $f^{\oplus}$

In this section, we define the computation of DNF formulas that are implied by the nth ancestor formula of an object s. Once again, the idea is to use approximations of the Anc(s, n) operator. Since we require that the approximations $f^{\oplus}$ are implied by the nth ancestor formula, it follows that each $f^{\oplus}$ formula we obtain will be satisfiable only by a superset of the nth ancestors of s. Roughly speaking, in order to obtain such formulas the idea is to rewrite each literal using a disjunction of literals obtained by choosing a single literal in each conjunction.

By changing the combinations of the chosen literals, we obtain different $f^{\oplus}$ formulas. Indeed, the idea is to derive the $f^{\oplus}$ formulas defining a new operator that mimics the operator Anc where at each rewriting step, each positive literal is replaced by a simple disjunction obtained starting from the DNF *cause* or *nocause*.

Formally, since there are many different $f^{\oplus}$ approximations of the nth ancestor formula, one for each possible choice of one literal in the conjunction, we define the operator computing $f^{\oplus}$ by means of a relation $\mathtt{Anc}^{\oplus}$ as follows.

Definition 15 Let $\mathcal{A} = (S, A)$ be a reaction systems and $s \in S$. We define a relation $\mathtt{Anc}^{\oplus} : S \times \mathbb{N} \times F_S$ where $\mathtt{Anc}^{\oplus}(s, n, f^{\oplus})$ if $\mathtt{Anc}^{\oplus}_{a}(s, n, f^{\oplus})$, where the auxiliary relation $\mathtt{Anc}^{\oplus}_{a} : F_S \times \mathbb{N} \times F_S$ is recursively defined as follows:

$$
\begin{array}{ll}
\mathtt{Anc}^{\oplus}_{a}(l,0,l) & \text{if } l = s \text{ or } l = \neg s\\
\mathtt{Anc}^{\oplus}_{a}(s,1,f^{\oplus}) & \text{if } \mathtt{Anc}^{\oplus}_{a}(cause(s),0,f^{\oplus}),\\
\mathtt{Anc}^{\oplus}_{a}(s,i,f^{\oplus}), i>1 & \text{if } \mathtt{Anc}^{\oplus}_{a}(\bigvee_{i\in\{1,\ldots,s\}} t_i, i-1, f^{\oplus}),\\
& cause(s) = \bigvee_{i\in\{1,\ldots,s\}} C_i,\\
& t_i \in Choose(C_i)\\
\mathtt{Anc}^{\oplus}_{a}(\neg s,1,f^{\oplus}) & \text{if } \mathtt{Anc}^{\oplus}_{a}(nocause(s),0,f^{\oplus}),\\
\mathtt{Anc}^{\oplus}_{a}(\neg s,i,f^{\oplus}), i>1 & \text{if } \mathtt{Anc}^{\oplus}_{a}(\bigvee_{i\in\{1,\ldots,s\}} t_i, i-1, f^{\oplus}),,\\
& nocause(s) = \bigvee_{i\in\{1,\ldots,s\}} C_i,\\
& t_i \in Choose(C_i)\\
\mathtt{Anc}^{\oplus}_{a}(f_1\vee f_2,i,f^{\oplus}_{1}\vee f^{\oplus}_{2}) & \text{if } \mathtt{Anc}^{\oplus}_{a}(f_1,i,f^{\oplus}_{1})\\
& \quad \text{and } \mathtt{Anc}^{\oplus}_{a}(f_2,i,f^{\oplus}_{2})\\
\mathtt{Anc}^{\oplus}_{a}(f_1\wedge f_2,i,f^{\oplus}_{1}\wedge f^{\oplus}_{2}) & \text{if } \mathtt{Anc}^{\oplus}_{a}(f_1,i,f^{\oplus}_{1})\\
& \quad \text{and } \mathtt{Anc}^{\oplus}_{a}(f_2,i,f^{\oplus}_{2})\\
\mathtt{Anc}^{\oplus}_{a}(true,i,true) & \\
\mathtt{Anc}^{\oplus}_{a}(false,i,false) &
\end{array}
$$

where $Choose(\bigwedge_{i\in\{1\ldots r\}} l_i) = \{l_1, l_2, \ldots, l_r\}$ and $\forall i \in \{1 \ldots r\}$, either $l_i = s$ or $l_i = \neg s$.

All formulas $f^{\oplus}$ in relation $\mathtt{Anc}^{\oplus}$ with s and n approximate the nth ancestor formula of s in the sense of Eq. (1). It is worth noting that, given an object of interest s, for any n, each $f^{\oplus}$ formula such that $\mathtt{Anc}^{\oplus}(s, n, f^{\oplus})$ is a DNF formula that is implied by $\mathtt{Anc}(s, n)$.

Theorem 6 *Let $\mathcal{A} = (S, A)$ be a reaction system. Let $f^{\oplus}$ such that $\mathtt{Anc}^{\oplus}(s, n, f^{\oplus})$ and $f = \mathtt{Anc}(s, n)$, for any object $s \in S$,*

- *$f^{\oplus}$ is a formula in disjunctive normal form,*
- *$f \to f^{\oplus}$.*

Proof Let us consider the following definition of the relation $\mathtt{Anc}^{\oplus}_{b}$ which differs from $\mathtt{Anc}^{\oplus}_{a}$ only for the case $\mathtt{Anc}^{\oplus}_{a}(s, 1, f^{\oplus})$. Indeed,

$$
\begin{array}{ll}
\mathtt{Anc}^{\oplus}_{b}(l,0,l) & \text{if } \mathtt{Anc}^{\oplus}_{a}(l,0,l)\\
\mathtt{Anc}^{\oplus}_{b}(s,i,f^{\oplus}),\ i>0 & \text{if } \mathtt{Anc}^{\oplus}_{a}(s,i,f^{\oplus})\\
\mathtt{Anc}^{\oplus}_{b}(\neg s,1,f^{\oplus}), i>0 & \text{if } \mathtt{Anc}^{\oplus}_{a}(\neg s,1,f^{\oplus})\\
\mathtt{Anc}^{\oplus}_{b}(f_1\vee f_2,i,f^{\oplus}_{1}\vee f^{\oplus}_{2}) & \text{if } \mathtt{Anc}^{\oplus}_{a}(f_1\vee f_2,i,f^{\oplus}_{1}\vee f^{\oplus}_{2})\\
\mathtt{Anc}^{\oplus}_{b}(f_1\wedge f_2,i,f^{\oplus}_{1}\wedge f^{\oplus}_{2}) & \text{if } \mathtt{Anc}^{\oplus}_{a}(f_1\wedge f_2,i,f^{\oplus}_{1}\wedge f^{\oplus}_{2})\\
\mathtt{Anc}^{\oplus}_{b}(true,i,true) & \\
\mathtt{Anc}^{\oplus}_{b}(false,i,false) &
\end{array}
$$

By using the same well-founded ordering of Theorem 5, we can prove that if f is a disjunction of literals and $\mathtt{Anc}^{\otimes}_{b}(f_b, n, f^{\oplus}_{b})$ then $f^{\oplus}_{b}$ is a disjunction of literals. The proof is analogous to the one of Theorem 5.

Now first observe that prove if $\mathtt{Anc}^{\oplus}_{b}(s, n, f^{\oplus}_{b})$ holds, then $f^{\oplus}_{b}$ is in DNF, since s is a literal. Now we want to prove that if $\mathtt{Anc}^{\oplus}_{a}(s, n, f^{\oplus})$ holds, then also $f^{\oplus}$ is in disjunctive normal form. The definition of $\mathtt{Anc}^{\oplus}_{a}$ differs from the one of $\mathtt{Anc}^{\oplus}_{b}$ only in one step, when $i = 1$. In this single step, literals can be replaced by a disjunction of conjunctions of literals. But if in a disjunction of literals we replace each literal with a disjunction of conjunctions of literals, we still obtain a formula in DNF. Thus, we can conclude that if $\mathtt{Anc}^{\oplus}(s, n, f^{\oplus})$ holds, then $f^{\oplus}$ is in DNF.

The proof of $f \rightarrow f^{\oplus}$ is trivial. If in a formula f we substitute some conjunction of subformulas with just one of the subformulas in the conjunction, we obtain a weaker formula which is implied by the original one, and this is what this approximation does by substituting each conjunction of literals in $cause(s)$ or in $nocause(s)$ with just one literal of the conjunction. □

Example 7 Consider again the simple reaction system $\mathcal{A}_2$ of Example 3. One approximation $f^{\oplus}$ of the 2nd ancestors of F is $(G \wedge B) \vee (H \wedge B)$, and it can be computed as follows.

$$
\begin{aligned}
\mathtt{Anc}^{\oplus}(F, 2, f^{\oplus}) \text{ if } & \mathtt{Anc}_a^{\oplus}(F, 2, f^{\oplus}) \\
\text{if } & \mathtt{Anc}_a^{\oplus}(D, 1, f^{\oplus})) \\
\text{if } & \mathtt{Anc}_a^{\oplus}((G \wedge B) \vee (H \wedge B), 0, f^{\oplus} = f_1^{\oplus} \vee f_2^{\oplus})) \\
\text{if } & \mathtt{Anc}_a^{\oplus}(G \wedge B, 0, f_1^{\otimes} = f_3^{\oplus} \wedge f_4^{\oplus}) \\
& \text{and } \mathtt{Anc}_a^{\otimes}(H \wedge B, 0, f_2^{\otimes} = f_4^{\oplus} \wedge f_5^{\oplus}) \\
\text{if } & \mathtt{Anc}_a^{\oplus}(G, 0, f_3^{\otimes} = G) \text{ and } \mathtt{Anc}_a^{\oplus}(B, 0, f_4^{\otimes} = B) \\
& \text{and } \mathtt{Anc}_a^{\otimes}(H, 0, f_4^{\otimes} = H) \\
& \text{and } \mathtt{Anc}_a^{\otimes}(B, 0, f_5^{\otimes} = B)
\end{aligned}
$$

Note that in $\mathtt{Anc}^{\oplus}(F, 2, f^{\oplus})$, $f^{\oplus} = (G \wedge B) \vee (H \wedge B)$ is already a DNF and that $\mathtt{Anc}(F, 2) = C \wedge ((G \wedge B) \vee (H \wedge B)) \rightarrow (G \wedge B) \vee (H \wedge B)$. This approximation of the 2nd ancestor formula that sets not containing B and containing neither G nor H cannot be 2nd ancestors of F. In particular, in this case, it assures us that the presence of B is *necessary* for a set to be the 2nd ancestors of F.

Of course, also C can be computed for the relation $\mathtt{Anc}^{\oplus}(F, 2, f^{\oplus})$ by choosing the other possible literal in the DNF $cause(F)$ formula. In that case, the approximation $f^{\oplus}$ assures us that the presence of C is *necessary* for a set to be the 2nd ancestors of F.

7.3 A biological application: discovering minimal ancestors for yeast cell cycle Boolean network

The cell cycle process by which a cell grows and divides into two cells is a vital process the regulation of which is conserved among the eukaryotes (Murray and Hunt 1993). The process mainly consists in the four phases depicted in Fig. 4.

In phase G1, the cell grows and, under appropriate conditions, commits to division; in phase S, the DNA is synthesized and chromosomes replicated: G2 is the phase where the cell checks the duplicated chromosomes; and finally, in the M (Mitosis) phase the cell is divided into two. After the Mitosis phase, the cell enters the G1 phase again, hence completing a "cycle." There are about 800 genes involved in the cell cycle process of the budding yeast (Spellman et al. 1998). However, the number of key regulators that are responsible for the control and regulation of this complex process is much smaller. Based on extensive literature studies, the authors in Li et al. (2004) constructed a network of key regulators involving 11 genes. The relations between genes are described by the Boolean network (M_{Cell}, E_{Cell}) depicted in Fig. 5, where the positive influence of a gene on another one is depicted in green, while the negative influence of a gene on another one is depicted in red. The Boolean network was used to study the time evolution of the protein states. Starting from the $2^{11} = 2048$ possible initial states describing a configuration for gene activation, they discover that all of them flow into one of the seven attractor stationary states. In particular, among the seven fixed points there is one big attractor that attracts 1764 initial states.

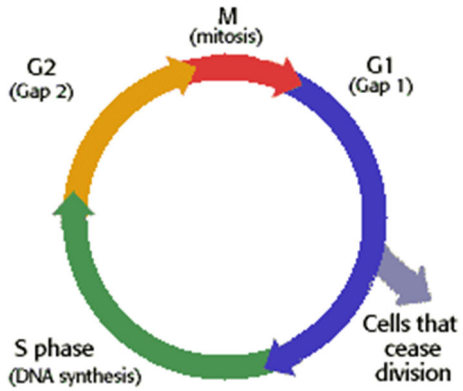

Fig. 4 Complete cell cycle

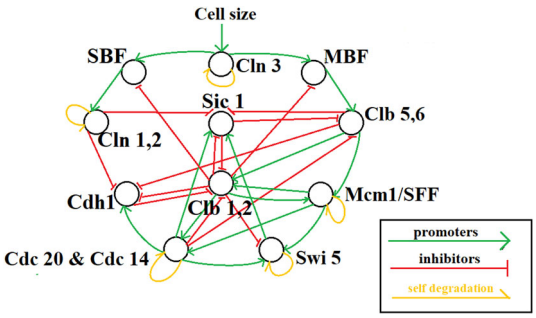

Fig. 5 Boolean network (M_{Cell}, E_{Cell})

In Barbuti et al. (2018b), some of these authors translated the Boolean network (M_{Cell}, E_{Cell}) of Fig. 5 into the following reaction system,

$\mathcal{A}_{cell} = (\{Cellsize, Cln3, Clb1,2, Clb5,5, SBF, MBF, Sic1, Swi5, Mcm1, Cdc20, Cdh1\}, \{a_1, \ldots, a_{50}\})$,

where the reactions $a_1 - a_{50}$ are listed in Fig. 6.

The execution of the reaction systems is able to mimic the 12 steps of evolution of activation/inactivation of genes of the Boolean network as it can be observed in nature and it is depicted in Fig. 7, where the activation of a gene is indicated with a dark circle.

At the beginning, the cell stays in a stationary state where only genes $Sic1$ and $Cdh1$ are active, while all the other genes are inactive. This is a stationary states and the cell waits for the external stimulus (the grow of the cell, here

$$
\begin{aligned}
a_1 &= (\{Cln3\}, \{Clb1,2\}, \{SBF, MBF\})\\
a_2 &= (\{SBF\}, \{Clb1,2\}, \{SBF\})\\
a_3 &= (\{SBF, Cln3\}, \emptyset, \{SBF\})\\
a_4 &= (\{MBF\}, \{Clb1,2\}, \{MBF\})\\
a_5 &= (\{MBF, Cln3\}, \emptyset, \{MBF\})\\
a_6 &= (\{SBF\}, \emptyset, \{Cln1,2\})\\
a_7 &= (\{Cdc20\}, \{Clb5,6, Clb1,2, Cln1,2\}, \{Sic1\})\\
a_8 &= (\{Swi5\}, \{Clb5,6, Clb1,2, Cln1,2\}, \{Sic1\})\\
a_9 &= (\{Sic1\}, \{Clb5,6, Clb1,2, Cln1,2\}, \{Sic1\})\\
a_{10} &= (\{Cdc20, Swi5\}, \{Clb1,2, Cln1,2\}, \{Sic1\})\\
a_{11} &= (\{Cdc20, Swi5\}, \{Clb5,6, Cln1,2\}, \{Sic1\})\\
a_{12} &= (\{Cdc20, Swi5\}, \{Clb5,6, Clb1,2\}, \{Sic1\})\\
a_{13} &= (\{Sic1, Cdc20, Swi5\}, \{Cln1,2\}, \{Sic1\})\\
a_{14} &= (\{Sic1, Cdc20, Swi5\}, \{Clb5,6\}, \{Sic1\})\\
a_{15} &= (\{Sic1, Cdc20, Swi5\}, \{Clb1,2\}, \{Sic1\})\\
a_{16} &= (\{Sic1, Cdc20, Swi5\}, \{Clb1,2\}, \{Sic1\})\\
a_{17} &= (\{Sic1, Cdc20\}, \{Clb1,2, Cln1,2\}, \{Sic1\})\\
a_{18} &= (\{Sic1, Cdc20\}, \{Clb5,6, Cln1,2\}, \{Sic1\})\\
a_{19} &= (\{Sic1, Cdc20\}, \{Clb5,6, Clb1,2\}, \{Sic1\})\\
a_{20} &= (\{Sic1, Swi5\}, \{Clb1,2, Cln1,2\}, \{Sic1\})\\
a_{21} &= (\{Sic1, Swi5\}, \{Clb5,6, Cln1,2\}, \{Sic1\})\\
a_{22} &= (\{Sic1, Swi5\}, \{Clb5,6, Clb1,2\}, \{Sic1\})\\
a_{23} &= (\{Cdc20\}, \{Clb5,6, Cln1,2, Clb1,2\}, \{Cdh1\})\\
a_{24} &= (\{Cdh1\}, \{Clb5,6, Cln1,2, Clb1,2\}, \{Cdh1\})\\
a_{25} &= (\{Cdh1Cdc20\}, \{Clb5,6, Clb1,2\}, \{Cdh1\})\\
a_{26} &= (\{Cdh1Cdc20\}, \{Cln1,2, Clb1,2\}, \{Cdh1\})\\
a_{27} &= (\{Cdh1Cdc20\}, \{Clb5,6, Cln1,2\}, \{Cdh1\})\\
a_{28} &= (\{MBF\}, \{Sic1, Cdc20\}, \{Clb5,6\})\\
a_{29} &= (\{MBF, Clb5,6\ \}, \{Cdc20\}, \{Clb5,6\})\\
a_{30} &= (\{MBF, Clb5,6\ \}, \{Sic1\}, \{Clb5,6\})\\
a_{28} &= (\{Clb5,6\}, \{Sic1, Cdc20\}, \{Clb5,6\})\\
a_{29} &= (\{Clb5,6\}, \emptyset, \{Mcm1\})\\
a_{30} &= (\{Clb1,2\}, \emptyset, \{Mcm1\})\\
a_{31} &= (\{Clb1,2\}, \emptyset, \{Cdc20\})\\
a_{32} &= (\{Mcm1\}, \emptyset, \{Cdc20\})\\
a_{33} &= (\{Cdc20\}, \{Clb12\}, \{Swi5\})\\
a_{34} &= (\{Mcm1\}, \{Clb12\}, \{Swi5\})\\
a_{35} &= (\{Cdc20, Mcm1\}, \emptyset, \{Swi5\})\\
a_{36} &= (\{Mcm1\}, \{Sic1, Cdh1, Cdc20\}, \{Clb1,2\})\\
a_{37} &= (\{Clb5,6\}, \{Sic1, Cdh1, Cdc20\}, \{Clb1,2\})\\
a_{38} &= (\{Clb1,2\}, \{Sic1, Cdh1, Cdc20\}, \{Clb1,2\})\\
a_{39} &= (\{Mcm1, Clb5,6\}, \{Cdh1, Cdc20\}, \{Clb1,2\})\\
a_{40} &= (\{Mcm1, Clb5,6\}, \{Cdh1, Sic1\}, \{Clb1,2\})\\
a_{41} &= (\{Mcm1, Clb5,6\}, \{Sic1, Cdc20\}, \{Clb1,2\})\\
a_{42} &= (\{Mcm1, Clb1,2\}, \{Cdh1, Cdc20\}, \{Clb1,2\})\\
a_{43} &= (\{Mcm1, Clb1,2\}, \{Cdh1, Sic1\}, \{Clb1,2\})\\
a_{44} &= (\{Mcm1, Clb1,2\}, \{Sic1, Cdc20\}, \{Clb1,2\})\\
a_{45} &= (\{Clb1,2, Clb5,6\}, \{Cdh1, Cdc20\}, \{Clb1,2\})\\
a_{46} &= (\{Clb1,2, Clb5,6\}, \{Cdh1, Sic1\}, \{Clb1,2\})\\
a_{47} &= (\{Clb1,2, Clb5,6\}, \{Sic1, Cdc20\}, \{Clb1,2\})\\
a_{48} &= (\{Clb1,2, Clb5,6, Mcm1\}, \{Cdh1\}, \{Clb1,2\})\\
a_{49} &= (\{Clb1,2, Clb5,6, Mcm1\}, \{Sic1\}, \{Clb1,2\})\\
a_{50} &= (\{Clb1,2, Clb5,6, Mcm1\}, \{Cdc20\}, \{Clb1,2\})
\end{aligned}
$$

Fig. 6 Reactions of $\mathcal{A}_{cell}$

indicated with the activation of gene $Cellsize$) to arrive. When also $Cellsize$ becomes active, this triggers the activation sequence of the genes, as shown in Fig. 7. After 12 steps, the cell completes its cycle and divides, going back to the initial stationary state waiting for another external stimulus to arrive and to make it divide again.

In this context, consider the (last) stationary state, that we will call S, where $Sic1$ and $Cdh1$ are active, while all the other genes are inactive. This can easily be done by adding the reaction
$(\{Sic1, Cdh1\}, \{Cellsize, Cln3, Clb1,2, Clb5,5, SBF, MBF, Swi5, Mcm1, Cdc20\}, \{S\})$.
Thus, we can obtain all the alternative ways to reach the stationary state S in n steps simply by computing the $n+1$th ancestor formula of S. For example,

$$
\begin{aligned}
\mathtt{Anc}(S,2) = {} & Cdh1 \wedge Swi_5 \wedge \neg Cdc20 \wedge \neg Clb1,2 \wedge \neg Clb5,6\\
& \wedge \neg Cln1,2 \wedge \neg Cln3 \wedge \neg MBF \wedge \neg Mcm1\\
& \wedge \neg SBF
\end{aligned}
$$

.

Note that in the previous formula positive literals correspond to genes that have to be active, while negative literals correspond to genes that have to be inactive for the stationary state S to be reached after one step. Of course, the last but one state of Fig. 7, which describes the yeast cell cycle that can be observed in nature, satisfies the formula $\mathtt{Anc}(S,1)$. The interesting point is that actually the stationary state S could be reached even if gene $Sic1$ was not active at the previous step. Indeed, the minimal 1st ancestor of S is $\{Cdh1, Swi5\}$ which does not contain $Sic1$. Computing the nth ancestor formula of S, we can discover all the (minimal) configurations of active/inactive genes that lead to the stationary state S. For example, computing the 2nd ancestor formula of S, we can discover that there are 16 alternative states (configurations) that lead to the stationary state S. However, increasing the number of steps increases the nesting level of AND–OR operators in the ancestor formula, and therefore, its conversion in DNF becomes more and more expensive. In these cases, it can be extremely useful to use the approximations introduced in the previous section since they allow the construction of approximation formulas with a lower complexity. Indeed, looking for a pair $f^{\otimes}$ and $f^{\oplus}$ such that the relations $\mathtt{Anc}^{\otimes}(S, 3, f^{\otimes})$ and $\mathtt{Anc}^{\oplus}(S, 3, f^{\oplus})$ hold allows us to compute the following approximations

$$
\begin{aligned}
f^{\otimes} = {} & Cdh20 \wedge Cdh1 \wedge \neg Clb12 \wedge \neg Clb1,2 \wedge \neg Clb5,6\\
& \wedge \neg Cln1,2 \wedge \neg Cln3 \wedge \neg MBF \wedge \neg Mcm1\\
& \wedge \neg SBF
\end{aligned}
$$

$$
f^{\oplus} = Cdh20 \vee Cdh20.
$$

The approximation $f^{\otimes}$ tells us that there is at least a 2nd ancestor of size 2; that is, if we start with both $Cdh20$ and $Cdh1$ active we can reach the S state in two steps. Note that this implies that neither the activation of $Sic1$ nor the one of $Swi5$ are really necessary for reaching the state S. From $f^{\oplus}$, on the contrary, we can conclude that the activation of $Cdh20$ is *necessary* for leading to the state S in two steps. A similar reasoning can be repeated for each backward step.

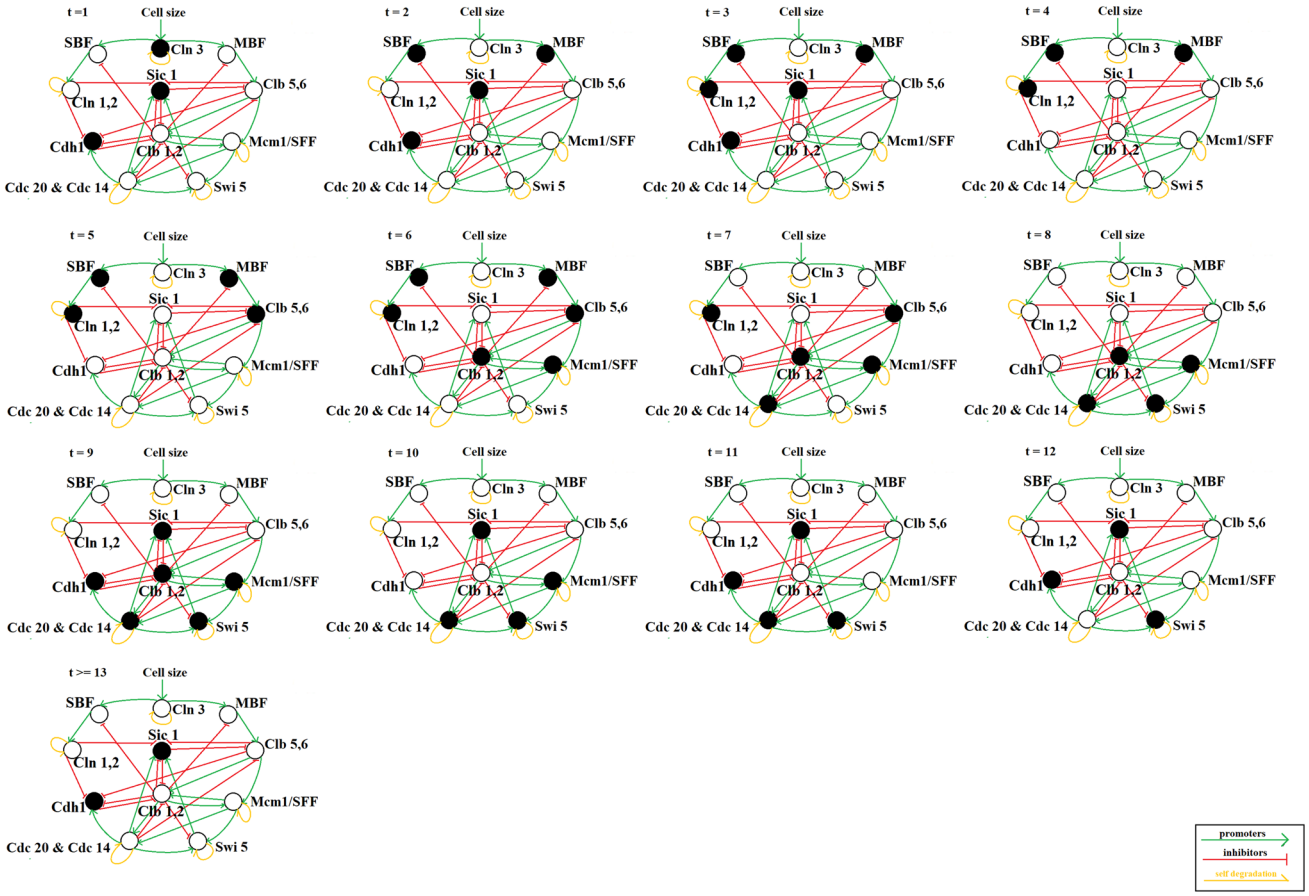

Fig. 7 Cell cycle evolution

Finally, computing a pair $f^{\otimes}$ and $f^{\oplus}$ such that the relations $\mathtt{Anc}^{\otimes}(S, 13, f^{\otimes})$ and $\mathtt{Anc}^{\oplus}(S, 13, f^{\oplus})$ hold will give us some information on the initial state that triggers all yeast cell cycle.

$$f^{\otimes} = Cln3 \wedge Sic1 \wedge \neg Clb1,2_1 \wedge \neg Clb5,6 \wedge \neg Cln1,2_1 \wedge \neg Mcm1 \wedge \neg SBF$$
$$f^{\oplus} = \neg Mcm1.$$

From $f^{\otimes}$, we can observe that the cell cycle could start even if $Cdh1$ is not active, while from $f^{\oplus}$ we can conclude that it is necessary that $Mcm1$ is not active.

8 Conclusions

In this paper, we have proposed a theory of ancestors for reactions systems, based on previously proposed notions of preimage/nth ancestors. An nth ancestor of a set of objects $s_1, \ldots, s_m$ is a set of objects that leads to the production of $s_1, \ldots, s_m$ in n evolution steps of reaction system. A preimage corresponds to a 1st ancestor.

Our theory is based on the new notion of nth ancestor formula, which is a logic formula that characterizes all the possible nth ancestors of a given object (or set of objects). We have defined an operator that computes nth ancestor formulas in polynomial time. This formula can be used to solve n ancestors problems in exponential time. The exponential time is actually due to the transformation of the formula into DNF. In order to tackle this problem, we proposed: (i) a syntactic restriction on the reaction system that ensures that the operator gives directly a formula in DNF and (ii) approximated notions of nth ancestor formulas that are directly computed in DNF and that provide a under- and a over-approximation of the set of all the nth ancestor of a given set of objects.

We have validated our approach by showing some real biological examples of gene regulatory networks where ancestors can provide useful information about the dynamics of the networks, even when the syntactic restrictions and the approximations we proposed are used.

As future work, we plan to apply our theory to investigate dynamical properties of more complex case studies of gene regulatory networks.

Acknowledgements Open access funding provided by Università di Pisa within the CRUI-CARE Agreement. This study was funded by the University of Pisa (grant PRA 2017_44, project "Metodologie informatiche avanzate per l'analisi di dati biomedici").

Compliance with ethical standards

Conflicts of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Barbuti R, Gori R, Levi F, Milazzo P (2016a) Investigating dynamic causalities in reaction systems. Theor Comput Sci 623:114–145

Barbuti R, Gori R, Levi F, Milazzo P (2016b) Specialized predictor for reaction systems with context properties. Fundamenta Informaticae 147(2–3):173–191

Barbuti R, Bernasconi A, Gori, R, Milazzo P (2018a) Computing preimages and ancestors in reaction systems. In: Theory and practice of natural computing—7th international conference (TPNC 2018), Dublin, Ireland, December 12–14, 2018, Proceedings, pp 23–35

Barbuti R, Bove P, Gori R, Levi F, Milazzo P (2018b) Simulating gene regulatory networks using reaction systems. In: Proceedings of the 27th international workshop on concurrency, specification and programming, Berlin, Germany, September 24–26, 2018

Barbuti R, Gori R, Levi F, Milazzo P (2018c) Generalized contexts for reaction systems: definition and study of dynamic causalities. Acta Informatica 55(3):227–267

Barbuti R, Gori R, Milazzo P (2018d) Predictors for flat membrane systems. Theor Comput Sci 736:79–102

Barbuti R, Gori R, Milazzo P (2017) Multiset patterns and their application to dynamic causalities in membrane systems. In: 18th International conference on membrane computing (CMC18). LNCS 10725. Springer, pp 54–73

Bodei C, Gori R, Levi F (2013) An analysis for causal properties of membrane interactions. Electron Notes Theor Comput Sci 299:15–31

Bodei C, Gori R, Levi F (2015) Causal static analysis for Brane calculi. Theor Comput Sci 587:73–103

Brayton RK, Sangiovanni-Vincentelli AL, McMullen CT, Hachtel GD (1984) Logic Minimization Algorithms for VLSI Synthesis. Kluwer Academic Publishers, Norwell, MA, USA

Brijder R, Ehrenfeucht A, Main MG, Rozenberg G (2011) A tour of reaction systems. Int J Found Comput Sci 22(7):1499–1517

Brijder R, Ehrenfeucht A, Rozenberg G (2010) A note on causalities in reaction systems. In: ECEASST, vol 30

Busi N (2007) Causality in membrane systems. In: Membrane computing, 8th international workshop (WMC 2007), Thessaloniki, Greece, June 25–28, 2007 Revised Selected and Invited Papers, pp 160–171

Cousot P, Cousot R (1977) Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Conference record of the fourth ACM symposium on principles of programming languages, Los Angeles, California, USA, January 1977, pp. 238–252

Cousot P, Cousot R (1979) Systematic design of program analysis frameworks. In: Conference record of the sixth annual ACM symposium on principles of programming languages, San Antonio, Texas, USA, January 1979, pp 269–282

Dennunzio A, Formenti E, Manzoni L, Porreca AE (2015a) Ancestors, descendants, and gardens of EDEN in reaction systems. Theor Comput Sci 608:16–26

Dennunzio A, Formenti E, Manzoni L, Porreca AE (2015b) Preimage problems for reaction systems. In: International conference on language and automata theory and applications (LATA 2015). Springer, pp 537–548

Ehrenfeucht A, Rozenberg G (2007) Reaction systems. Fundamenta informaticae 75(1–4):263–280

Elowitz MB, Leibler S (2000) A synthetic oscillatory network of transcriptional regulators. Nature 403(6767):335

Formenti E, Manzoni L, Porreca AE (2014) Fixed points and attractors of reaction systems. In: Conference on computability in Europe, pp 194–203

Gori R, Levi F (2010) Abstract interpretation based verification of temporal properties for bioambients. Inf. Comput. 208(8):869–921

Hassoun S, Sasao T (eds) (2002) Logic Synthesis and Verification. Kluwer Academic Publishers, Dordrecht

Li F, Long T, Lu Y, Ouyang Q, Tang C (2004) The yeast cell-cycle network is robustly designed. Proc Natl Acad Sci 101(14):4781–4786

Murray A, Hunt T (1993) The Cell Cycle. Oxford Univ. Press, New York

Salomaa A (2013) Functional constructions between reaction systems and propositional logic. Int. J. Found. Comput. Sci. 24(1):147–160

Salomaa A (2013) Minimal and almost minimal reaction systems. Nat Comput 12(3):369–376

Spellman P, Sherlock G, Zhang M, Iyer V, Anders K, Eisen M, Brown P, Botstein D, Futcher B (1998) Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. Mol Biol Cell 9(12):3273–3297

Umans C, Villa T, Sangiovanni-Vincentelli AL (2006) Complexity of two-level logic minimization. IEEE Trans CAD Integ Circuits Systems 25(7):1230–1246

Vilar JM, Guet CC, Leibler S (2003) Modeling network dynamics: the lac operon, a case study. J Cell Biol 161(3):471–476

Wong P, Gladney S, Keasling JD (1997) Mathematical model of the lac operon: inducer exclusion, catabolite repression, and diauxic growth on glucose and lactose. Biotechnol Prog 13(2):132–143

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.